file **server** and stored in a RAM location for execution. The context of the container **applet** which contains the container's handle and database **identification** is registered with context manager 21 in step S902. After the container context has been registered and the container created, an editor **applet** is selected and executed in step S903. At this point, the editor **applet** examines the context of the current container which is stored in context manager 21. In step S903, the editor applet. . .

=> d his

```
     (FILE 'USPAT' ENTERED AT 07:08:26 ON 21 OCT 1998)
L1             0 S SERVLET
L2             0 S SERVLETS
L3            18 S ACTIVEX
L4            11 S SERVER AND L3
L5          8602 S DOWNLOAD?
L6             9 S L4 AND L5
L7        151323 S SERVICE
L8             6 S L7 AND L6
L9           951 S SUN MICROSYSTEM? /ASN
L10          234 S JAVA
L11           27 S L9 AND L10
L12           14 S HOTJAVA
L13            9 S L11 AND L12
L14            7 S L5 AND L13
L15           97 S APPLET#
L16            3 S L15 AND L14
L17       125112 S (USER OR IDENTIF? OR AUTHENTIC?)(30A)(SERVER OR REMOTE O
R C
L18        37367 S L17(50A)(CLASS? OR SERVICE OR PROGRAM OR LOAD? OR DOWNLO
AD?
L19           25 S APPLET#(50A)(L18)
```

=> d kwic

US PAT NO: 5,828,840 [IMAGE AVAILABLE] L5: 1 of 1
TITLE: **Server** for starting **client** application on **client** if **client** is network terminal and **initiating client** application on **server** if **client** is non network terminal

ABSTRACT:
A plurality of clients are **connected** to one or more servers. When a **client initiates** a **connection** with a **server**, the **server** responds to the request for **connection** by transmitting a message back to the **client** to determine whether the **client** is a network terminal or not. The **client** responds with a message that is received by an application dispatcher at the **server** which takes one of a pair of actions based on whether the **client** is a network terminal. If the **client** terminal is a network terminal, then the application dispatcher spawns a **server** application in the **server** which responds to the **client** application in the **client**. Going forward, the **server** application responds to all future requests from the **client** application. If the **client** is not a network terminal, then the application dispatcher **initiates** a **client** application in the **server** to **service** the **client** terminal application requirements. Requests from the **client** application on behalf of the **client** terminal are subsequently serviced by a **server** application at the **server** which communicates to the **client** terminal via the **client** application at the **server**.

SUMMARY:

BSUM(4)

This invention generally relates to improvements in computer systems, and more particularly, to system software for managing a network of heterogeneous **client** terminals communicating with a **server** in a consistent manner.

SUMMARY:

BSUM(6)

Recently, . . . autonomy or processing ability which allows it to perform certain tasks without assistance from the mainframe to which it is **connected**. Many such devices are programmable by virtue of including a microprocessor.

SUMMARY:

BSUM(7)

While . . . as a terminal if a user interacts with the device to communicate to a host processor, referred to as a **server** in a network computing environment. Examples of terminals include keyboard/printer terminals, cathode-ray tube (CRT) terminals, remote-batch terminals, real-time data-acquisition and. . .

SUMMARY:

BSUM(8)

A . . . block transmissions, either on host or human command. If the terminal contains a microprocessor which runs a standard program to **service** the terminal, and not arbitrary, user-loaded programs, the terminal has a fixed function, and is still just an intelligent terminal.. . .

SUMMARY:

BSUM(10)

The Java language solves many of the **client**-side problems by:

SUMMARY:

BSUM(14)

Java is compiled into bytecodes in an intermediate form instead of machine **code** (like C, C++, Fortran, etc.). The bytecodes execute on any machine with a Java bytecode interpreter. Thus, Java applications can run on a variety of **client** machines, and the bytecodes are compact and designed to transmit efficiently over a network which enhances a preferred embodiment with universal clients and **server**-centric policies.

SUMMARY:

BSUM(15)

With . . . can create robust User Interface (UI) components. Custom "widgets" (e.g. real-time stock tickers, animated icons, etc.) can be created, and **client**-side performance is improved. Unlike HTML, Java supports the notion of **client**-side validation, offloading appropriate processing onto the **client** for improved performance. Dynamic, real-time applications can be created using the above-mentioned components.

SUMMARY:

BSUM(16)

Sun's . . . documents (e.g. simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g. Netscape Navigator) by copying **code** from the **server** to **client**. From a language standpoint, Java's core feature set is based on C++. Sun's Java literature states that Java is basically. . .

SUMMARY:

BSUM(17)

A . . . accordance with a preferred embodiment would execute Java applications in stand-alone mode, but have the capability to interact with a **server** for such functions as retrieving information, database processing, massive computation processing and access to shared devices such as high-speed printers,. . .

SUMMARY:

BSUM(22)

Fourth, . . . compiler for the particular terminal) to a mail program that is compatible with every different terminal attached to the host **server**. Some applications can be rebuilt for a particular terminal by simply recompiling the application, but many are only distributed as. . .

SUMMARY:

BSUM(24)

The . . . overcome in an illustrative embodiment of the invention in a network computing environment in which a plurality of clients are **connected** to one or more servers. When a **client initiates** a **connection** with a **server**, the **server** responds to the request for **connection** by transmitting a message back to the **client** to determine whether the **client** is a network terminal or not. The **client** responds with a message that is received by an application dispatcher at the **server** which takes one of a pair of actions based on whether the **client** is a network terminal. If the **client** terminal is a network terminal, then the application dispatcher spawns a **server** application in the **server** which responds to the **client** application in the **client**. Going forward, the **server** application responds to all future requests from the **client** application. If the **client** is not a network terminal, then the application dispatcher **initiates** a **client** application in the **server** to **service** the **client** terminal application requirements. Requests from the **client** application on behalf of the **client** terminal are subsequently serviced by a **server** application at the **server** which communicates to the **client** terminal via the **client** application at the **server**.

DRAWING DESC:

DRWD(4)

FIG. 2 illustrates a **client-server** network in accordance with a preferred embodiment;

DRAWING DESC:

DRWD(5)

FIG. 3 illustrates a **server** architecture in accordance with a preferred embodiment;

DRAWING DESC:

DRWD(6)

FIG. 4 illustrates a **client-server** architecture in accordance with a preferred embodiment;

DRAWING DESC:

DRWD(7)

FIG. 5 illustrates a first **client** request to a **server** in accordance with a preferred embodiment;

DRAWING DESC:

DRWD(8)

FIG. 6 illustrates a **client server** environment which accesses support services in accordance with a preferred embodiment;

DRAWING DESC:

DRWD(9)

FIG. 7 is an architecture diagram of a **client-server** system in

accordance with a preferred embodiment;

DRAWING DESC:

DRWD(10)

FIG. 8 is an architecture diagram of a **client-server** system in accordance with a preferred embodiment;

DRAWING DESC:

DRWD(11)

FIG. 9 is an architecture diagram of a **client-server** system in accordance with a preferred embodiment;

DETDESC:

DETD(2)

The . . . HP, or a Windows NT computer. A representative hardware environment is depicted in FIG. 1, which illustrates a typical hardware **configuration** of a computer 100 in accordance with the subject invention. The computer 100 is controlled by a central processing unit. . . computer may only have some of the units illustrated in FIG. 1, or may have additional components not shown, most **server** computers will include at least the units shown.

DETDESC:

DETD(3)

Specifically, . . . memory (RAM) 106 for temporary storage of information, a read only memory (ROM) 104 for permanent storage of the computer's **configuration** and basic operating commands and an input/output (I/O) adapter 110 for **connecting** peripheral or network devices such as a disk unit 113 and printer 114 to the bus 108, via cables 115 or peripheral bus 112, respectively. A user interface adapter 116 is also provided for **connecting** input devices, such as a keyboard 120, and other known interface devices including mice, speakers and microphones to the bus 108. Visual output is provided by a display adapter 118 which **connects** the bus 108 to a display device 122, such as a video monitor. The computer has resident thereon and is. . .

DETDESC:

DETD(4)

FIG. 2 illustrates a **client-server** network in accordance with a preferred embodiment. A set of consumer devices (**client** terminals 200) are attached to a **server** 210 and the **server** is attached to a legacy host 220 to process applications requiring information at the host 220. The **connection** could be by means of the Internet, a dialup link, token ring, cellular phone, satellite, T1 or X.25 telco link. . .

DETDESC:

DETD(5)

Server Software

DETDESC:

DETD(6)

The . . . using a combination of Java, C or possibly C++. C or C++

will be used mainly to implement platform dependent **code** (such as dealing with the comm[redacted]ts). While a preferred embodim[redacted] discloses support for a dial up network and Internet. . .

DETDESC:

DETD(8)

A **server** architecture in accordance with a preferred embodiment supports two types of **client** terminals.

DETDESC:

DETD(10)

These are **client** terminals capable of directly executing the Java applications on the **client** terminal which are **initially** stored on a **server**. The **server** will simply **download** this **code** to the **client**'s network terminal which the **client** will then execute to provide a particular **service**. This **service** may or may not interact with other clients or servers. Network terminals can be **connected** to a **server** through a dial up modem link, directly through a local area network, or by other network communication means in accordance. . .

DETDESC:

DETD(12)

These are **client**'s terminals which are not capable of executing Java applications on the **client** terminal. When dealing with this class of **client** the **server** will execute the application on behalf of the **client**. In this case the **server** will only expect necessary input and output operations to be performed by the **client** terminal. An example of how to **connect** a plurality of non-network terminals to a host **server** is described in U.S. Pat. No. 5,287,461, the disclosure of which is hereby incorporated by reference in its entirety.

DETDESC:

DETD(13)

FIG. 3 illustrates a **server** architecture in accordance with a preferred embodiment. A **client** 300 would **initiate** a **connection** with a **server** 350 by, for example, dialing in to a modem pool which is intercepted by the point-to-point stack software 311 which conforms information received to the TCP layer 312 which obtains a socket 313 for **connecting** the **client** 310 to the **server** 350. The Java net layer 314 further refines the request to conform with the TERMIO and NET layer 315 which passes the request along to the application dispatcher 319. The application dispatcher 319 spawns the appropriate **server** application selected from the **server** applications 330. On a non-network terminal, The non-network terminal **initiates** a "first **connection**" by dialing up a modem, for example. The dial up goes through the native OS 316 (Solaris or Windows NT dial up layer) and is **connected** with the serial communication in the VFI.SERIAL layer 317 which abstracts the serial input/output functions into a higher level communication. . . and maps it into a similar communication as the communication from the network terminal 300. It makes the dialup asynchronous **connection** appear to the **server** application as a new socket **connection**.

DETDESC:

DETD(14)

Network Terminal--"First **Connection**"

DETDESC:

DETD(15)

FIG. 4 illustrates a **client-server** architecture in accordance with a preferred embodiment. The architecture is illustrated **initially** for a network terminal for clarity and then follows with a non-network terminal. Processing commences at 400 when a network terminal requests **connection** through a layered communication system to a set of **server** threads 420 which are triggered by a detection of a "ring" 430 to **initiate** possible **client** updates and the subsequent **client** appplication to **server** application processing. "Ring" refers to a "first **connection**" in socket processing in accordance with a preferred embodiment.

DETDESC:

DETD(16)

The network terminal makes its **connection** through the Point-to-Point-Protocol stack 411 utilizing the TCP layer 412 and the sockets layer 413, which is like an electrical. . . terminals to communication sockets to facilitate communication through the network. All of this is managed by the Java.net 414 which **connects** the socket 1111 via the TCP layer 412 and the PPP stack 411. The layer above is the VFI.net and VFI.TERMIO 415 which is responsible for detecting that the **connection** is made and mapping the **connection** to an application dispatcher 431 to further process the first **connection** (ring) request.

DETDESC:

DETD(17)

The **server** 450 waits for a "first **connection**" request much like an interrupt manager. When a "first **connection**" request arrives, then the application dispatcher has a method that detects a **connect** request or a LAN "first **connection**" request that would arrive through the TCP layer as a socket **connect**. That **connection** is translated into a logical ring which is equivalent to an event or interrupt. The **server** 450 responds to the "first **connection**" with a query **initiated** by the application dispatcher 431 requesting "who are you" via an enquiry message asking for **identification** by the **client** loader thread 421. The network terminal responds with **ID** information, including the **identification** of the application that the network terminal requires. If the terminal answers with an **identifier** indicating that the terminal is a network terminal, then the **client** loader thread 421 performs any necessary **client** application updates via a **download** using a file transfer program such as UDP or FTP, or any other socket layer protocols that are available for. . .

DETDESC:

DETD(18)

Network Terminal--First **Client** Request to **Server**

DETDESC:

DETD(19)

FIG. 5 illustrates a first **client** request to a **server** in accordance with a preferred embodiment. When a first **client** request is transmitted from the network terminal 500 with a **client**

application resident thereon 510 to the **server** 550, the application dispatcher 530 spawns a corresponding **server** application 520 for servicing the request to the **server** 550 via the assigned socket 1112. The **server** application 520 responds to the request and transmits information to the network terminal 500. The application dispatcher 530 has completed its responsibilities for this **client** 500 and can return to a wait state until the next "first **connection**" request from a **client**. The **client** application request could be as simple as a get current time request or a request for data from a **server** database.

DETDESC:

DETD(20)

Network Terminal--Subsequent **Client** Request to **Server**

DETDESC:

DETD(21)

FIG. 6 illustrates a network terminal 600 with a **downloaded client** application 610 which accesses support services in the **server** 650 through its assigned **server** application 620 in accordance with a preferred embodiment. The terminal 600 communicates to a **server** application 620 which accesses host processing capabilities and database services 640 to **service** requests emanating from the **client** application 610. The **server** application 620 handles any events that originate from the **client** application 610 via the assigned socket 1112. These events could include data requests from a database application, or data transfer to a **server**. Remote data from another **server** application could also be accessed by the **client**. **Server** application 620 accesses support services directly or via a socket interface 660.

DETDESC:

DETD(22)

Non-network Terminal--"First **Connection**"

DETDESC:

DETD(23)

FIG. 7 is an architecture diagram of a **client-server** system in accordance with a preferred embodiment. A layered communication system 700 is used by a non-network terminal 710 to detect a ring providing an indicia of communication 740 and dispatch an application 730. Dispatching an application 730 also **initiates** a **server** thread 720 for servicing the **client** request. The non-network terminal 710 **initiates** a "first **connection**" by dialing up a modem, for example. The dial up goes through the native OS 711 (Solaris or Windows NT dial up layer) and is **connected** with the serial communication in the VFI.SERIAL layer 712 which abstracts the serial input/output functions into a higher level communication. . . layer and maps it into a similar communication as the communication from the network terminal. It makes the dialup asynchronous **connection** appear to the **server** application as a new socket **connection** 1111. The communication is an event 740 that triggers actions by the application dispatcher 741 which responds to the "first **connection**" event by requesting **ID** information from the **client**, via an enquiry message, and starting the requested **client** application 720 at the **server** 750.

DETDESC:

DETD(24)

Non-network Terminal--First **Client** Request To **Server**

DETDESC:

DETD(25)

FIG.8 is an architecture diagram of a **client-server** system in accordance with a preferred embodiment. The **client** application 822 is responsible for managing the non-network terminal 810. The **client** application 822 writes information, utilizing a **server** version of VFI.TERMIO 855, to and responds to key presses by the non-network terminal 810 at the **server** 850. The **client** application 822 **initially** makes a request for **service** from a socket 1112 that is associated with the non-network terminal 810 when the application dispatcher 840 spawns the **client** application 822.

DETDESC:

DETD(26)

When the first request 845 is generated by the **client** application 822 residing on the **server** 850, at application startup, the first request for **service** is routed in the **server** 850 to the application dispacher 840 and spawns the **server** application 820 which will handle subsequent requests. The **server** application 820 makes a request for **service** from a socket 1112 that is associated with the **client** application 822 which transmits an appropriate command through the VFI.TERMIO 855 to the VFI.SERIAL layer 856 using the operating system. . . terminal 810. This processing is identical to the network terminal processing with the exception that all applications reside on the **server** 850 as opposed to a Java application executing remotely on the network terminal.

DETDESC:

DETD(27)

One advantage of Java is that it is machine independent and does not care whether a Java application resides on the **client** or the **server**. In the case of the non-network terminal, the **client** application resides in the **server** and controls the java incapable terminal.

DETDESC:

DETD(28)

Non-network Terminal--Subsequent **Client** Requests to **Server**

DETDESC:

DETD(29)

FIG. 9 is an architecture diagram of a **client-server** system in accordance with a preferred embodiment. A layered communication system 900 is used by a non-network terminal 910 to manage the interconnection of a **server** Application 940 to a **client** application 920 and facilitate communication between the terminal 910 and **server** application 940 via a **client** application 920 resident on the **server** 950. FIG. 9 shows the processing after the first request has been completed and the **client** application 920 is coupled with the **server** application 940 via the assigned socket 1112 just as in the network terminal example, except the **client** application 920 and

server application 940 both reside on the **server** 950.

DETDESC:

DETD(30)

If . . . supported with the command streams described in FIGS. 10-14. If the terminal is a network terminal, then the application is **downloaded** via a FTP or other network file transfer procedure.

DETDESC:

DETD(32)

FIG. 12 represents a table showing additional details associated with the device types, commands and data **parameters**. For example, the device type field is one byte long and specifies the selected Input/Output device. FIG. 13 illustrates the. . .

DETDESC:

DETD(33)

FIG. . . . communication flow diagram in accordance with a preferred embodiment. A terminal 1500 either has firmware or an application 1504 that **initiates** a **connection** 1506 with a **server** 1502 by contacting a dispatcher 1508. The **connect initiate** 1506 also **connects** a socket 1111 to handle the **connection**. The dispatcher 1508 transmits an **identification** enquiry 1510 which the **client** terminal replies to with an **identification** message 1512. In the case of a network terminal, the **client** loader 1522 performs any necessary **client** application updates 1520 on the **client** terminal 1500. In the case of a non-network terminal, the dispatcher starts the **client** application. The **client** then sends a request to start the **server** application 1530 to the **server** which results in the **connection** of a socket 1112 and the **server** application 1550 being started and a confirmation message 1532 being transmitted back to the **client** application 1540. Then, when the **client** application 1540 requests data 1542 from the **server** application 1550, the **server** application 1550 responds with the application response data 1560.

DETDESC:

DETD(35)

Configured modem ports that will take part in transactions are pre-**configured**. The Application Dispatcher (AD) startup **code** looks at this **configuration** stream to determine the number of S threads (serial port listeners). S classes instantiate a VFI.NET.serversocket object which in turn create a VFI.NET.ModemIO.ModemPort object. The ModemPort object binds to a low level VFI.NET.ModemIO.Port object which utilizes native methods to **configure** and wait on the communications port.

DETDESC:

DETD(36)

```
SO
serversocket SOSocket = new serversocket ("socket1111", 1);
//Listener object
{
socket SOConnSocket= SOSocket.accept( ); //
Translates to
WaitDevice(CONNECT)
```

```
ReadAndValidate (RequestID);
return RequestID, SOCKET.Socket;
}
}
```

DETDESC:

DETD(42)

The AD will query a database to determine which applications should be **initiated** based on the enquiry message utilizing an SQL query of the form:

DETDESC:

DETD(43)
"SELECT<Field . . . is handled by the JDBC layers to return data to the AD. The AD is now ready to run the **client** thread.

DETDESC:

DETD(45)

The field list contains appropriate fields (those required for **client** application processing) and are passed down to the **client** thread along with the **connected** socket object.

DETDESC:

DETD(46)

Client Threads

DETDESC:

DETD(47)

Client Threads **proxy** the actual application. Application output meant for the terminal's devices are routed out using VFI.TERMIO as directives to the **client** terminal's firmware. The **connected** socket (which translates to a live dial-up **connection**) is passed down from the AD to the **client** thread. **Client** threads are long living--usually transferring data to corresponding servlets that **initiate connections** to upstream hosts or make database transactions. Despite the fact that **client** threads can be JDBC aware, servlets handle database transactions. This helps to maintain **code** constancy when the same **client** class is **downloaded** to a Java capable terminal for remote execution.

DETDESC:

DETD(48)

Terminal . . . a VFI.TermIO object that in turn instantiates a VFI.TermIO.ServProtocol object. The protocol object implements the actual data transfer with the **client** terminal. The protocol object requires the socket object passed down from the AD to the **client** thread.

DETDESC:

DETD(49)

```
. . .
      //instantiation. This cascades into a ServProtocol
Object instantiation.
IOObject.WriteString (StringIndex); //Displays a particular
```

references to externally located program files; Arthur A. van Hoff, 395/200.3, 200.42, 20[redacted]7 [IMAGE AVAILABLE]

=> d 1-9 fd,rel

US PAT NO: 5,802,530 [IMAGE AVAILABLE] L13: 1 of 9
DATE FILED: Jul. 1, 1996

US PAT NO: 5,794,049 [IMAGE AVAILABLE] L13: 2 of 9
DATE FILED: Jun. 5, 1996

US PAT NO: 5,790,855 [IMAGE AVAILABLE] L13: 3 of 9
DATE FILED: Jan. 31, 1997

US PAT NO: 5,765,157 [IMAGE AVAILABLE] L13: 4 of 9
DATE FILED: Jun. 5, 1996

US PAT NO: 5,761,513 [IMAGE AVAILABLE] L13: 5 of 9
DATE FILED: Jul. 1, 1996

US PAT NO: 5,761,421 [IMAGE AVAILABLE] L13: 6 of 9
DATE FILED: Mar. 25, 1996

US PAT NO: 5,754,857 [IMAGE AVAILABLE] L13: 7 of 9
DATE FILED: Dec. 8, 1995

US PAT NO: 5,736,984 [IMAGE AVAILABLE] L13: 8 of 9
DATE FILED: Jul. 3, 1996

US PAT NO: 5,727,147 [IMAGE AVAILABLE] L13: 9 of 9
DATE FILED: Dec. 8, 1995

=> d his

(FILE 'USPAT' ENTERED AT 07:08:26 ON 21 OCT 1998)
L1 0 S SERVLET
L2 0 S SERVLETS
L3 18 S ACTIVEX
L4 11 S SERVER AND L3
L5 8602 S DOWNLOAD?
L6 9 S L4 AND L5
L7 151323 S SERVICE
L8 6 S L7 AND L6
L9 951 S SUN MICROSYSTEM? /ASN
L10 234 S JAVA
L11 27 S L9 AND L10
L12 14 S HOTJAVA
L13 9 S L11 AND L12

=> s l5 and l13

L14 7 L5 AND L13

=> d 1-7

1. 5,802,530, Sep. 1, 1998, Web document based graphical user interface; Arthur A. Van Hoff, 707/513; 345/335; 395/200.33, 682 [IMAGE AVAILABLE]

2. 5,794,049, Aug. 11, 1998, Computer system and method for executing architecture specific code with reduced run-time memory space requirements; Timothy G. Lindholm, 395/706, 704, 709 [IMAGE AVAILABLE]

3. 5,790,855, Aug. 4, 1998, System, method and article of manufacture for type checking appropriateness of port connection and variable type matching in connection with multiport object-oriented components; Antony Azio Faustini, 395/701; 345/348 [IMAGE AVAILABLE]

4. 5,765,157, Jun. 9, 1998, Computer system and method for executing threads of execution with reduced run-time memory space requirements; Timothy G. Lindholm, et al., 707/101; 341/51, 79; 707/205, 206; 711/113, 160 [IMAGE AVAILABLE]

5. 5,761,421, Jun. 2, 1998, System and method for secure peer-to-peer communication between **downloaded** programs; Arthur A. van Hoff, et al., 395/200.53; 340/825.5; 395/200.58 [IMAGE AVAILABLE]

6. 5,754,857, May 19, 1998, Distributed asynchronous workflow on the net; Steven D. Gadol, 395/680, 684 [IMAGE AVAILABLE]

7. 5,727,147, Mar. 10, 1998, System and method for resolving symbolic references to externally located program files; Arthur A. van Hoff, 395/200.3, 200.42, 200.47 [IMAGE AVAILABLE]

=> s applet#

L15 97 APPLET#

=> s l15 and l14

L16 3 L15 AND L14

=> d 1-3

1. 5,802,530, Sep. 1, 1998, Web document based graphical user interface; Arthur A. Van Hoff, 707/513; 345/335; 395/200.33, 682 [IMAGE AVAILABLE]

2. 5,790,855, Aug. 4, 1998, System, method and article of manufacture for type checking appropriateness of port connection and variable type matching in connection with multiport object-oriented components; Antony Azio Faustini, 395/701; 345/348 [IMAGE AVAILABLE]

3. 5,761,421, Jun. 2, 1998, System and method for secure peer-to-peer communication between **downloaded** programs; Arthur A. van Hoff, et al., 395/200.53; 340/825.5; 395/200.58 [IMAGE AVAILABLE]

=> d 1-3 fd,rel

US PAT NO: 5,802,530 [IMAGE AVAILABLE] L16: 1 of 3
DATE FILED: Jul. 1, 1996

US PAT NO: 5,790,855 [IMAGE AVAILABLE] L16: 2 of 3
DATE FILED: Jan. 31, 1997

US PAT NO: 5,761,421 [IMAGE AVAILABLE] L16: 3 of 3
DATE FILED: Mar. 25, 1996

=> d 1,3 ab,kwic

US PAT NO: 5,802,530 [IMAGE AVAILABLE] L16: 1 of 3

ABSTRACT:
A Web document based GUI for use on a client computer that is networked with server computers. The GUI enables a user of the client computer to initiate specific operations that are performed on the client computer

and that define a particular application. The GUI comprises GUI Web documents and a Web browser. Each GUI Web document is located at the client computer or one of the server computers and comprises one or more links and one or more **applets**. Each link provides a link to a corresponding GUI document when selected by the user with the client computer while being displayed on the client computer. Each respective **applet** generates, when executed on the client computer, an interactive image that is displayed on the client computer. The user can initiate a respective operation (i.e., one of the GUI's specific operations) by acting on the interactive image with the client computer to invoke the respective **applet** to perform the respective operation on the client computer. The Web browser runs on the client computer and, each time a displayed link of a displayed GUI Web document has been selected by the user with the client computer, loads in, if not already loaded, and displays on the client computer the corresponding GUI Web document. The Web browser displays the corresponding GUI web document by executing each of the one or more **applets** of the corresponding GUI Web document and displaying on the client computer the corresponding interactive image and by displaying on the client computer the one or more links of the corresponding GUI Web document. The Web browser comprises an editor that edits on the client computer certain GUI Web Documents by adding and/or removing **applets** and links from the certain GUI Web documents. In this way, the GUI can be customized.
ASSIGNEE: **Sun Microsystems**, Inc., Palo Alto, CA (U.S. corp.)

ABSTRACT:
A . . . at the client computer or one of the server computers and comprises one or more links and one or more **applets**. Each link provides a link to a corresponding GUI document when selected by the user with the client computer while being displayed on the client computer. Each respective **applet** generates, when executed on the client computer, an interactive image that is displayed on the client computer. The user can. . . one of the GUI's specific operations) by acting on the interactive image with the client computer to invoke the respective **applet** to perform the respective operation on the client computer. The Web browser runs on the client computer and, each time. . . GUI Web document. The Web browser displays the corresponding GUI web document by executing each of the one or more **applets** of the corresponding GUI Web document and displaying on the client computer the corresponding interactive image and by displaying on. . . The Web browser comprises an editor that edits on the client computer certain GUI Web Documents by adding and/or removing **applets** and links from the certain GUI Web documents. In this way, the GUI can be customized.

SUMMARY:

BSUM(7)

The . . . by the WWW guarantee that any Web browser can communicate with any Web server. However, until the invention of the **Java** programming language and **Java applets** (i.e., programs written in the **Java** programming language that are part of a Web document), there was no way to provide platform independent programs over the. . .

SUMMARY:

BSUM(8)

An important feature of the **Java** programming language is the platform independence of **Java applets** written in the **Java** language and compiled into **Java** bytecode. This means that such programs can be executed on any computer having a **Java** virtual machine module where the **Java** virtual machine module interprets the **Java applets** for execution on the specific platform of the computer.

SUMMARY:

BSUM(9)

Another important feature of **Java applets** is the verifiability of their integrity by a **Java** virtual machine module prior to their execution. The **Java** virtual machine module determines whether **Java applets** conform to predefined stack usage and data usage restrictions to ensure that **Java applets** cannot overflow or underflow the virtual machine module's stack and utilize only data of known data types. As a result, **Java applets** cannot create object pointers and generally cannot access system resources other than those resources which the user explicitly grants it permission to use. Consequently, when **Java applets** are **downloaded** to a client computer, a Web browser that is running on the client computer and has a **Java** virtual machine module will be able to verify and then execute the **downloaded applets**.

SUMMARY:

BSUM(13)

Each . . . at the client computer or one of the server computers and comprises one or more links and one or more **applets**. Each link provides a link to a corresponding GUI document when selected by the user with the client computer while being displayed on the client computer. Each respective **applet** generates, when executed on the client computer, an interactive image that is displayed on the client computer. The user can. . . one of the GUI's specific operations) by acting on the interactive image with the client computer to invoke the respective **applet** to perform the respective operation on the client computer.

SUMMARY:

BSUM(14)

The . . . GUI Web document. The Web browser displays the corresponding GUI web document by executing each of the one or more **applets** of the corresponding GUI Web document and displaying on the client computer the corresponding interactive image and by displaying on. . .

SUMMARY:

BSUM(15)

The Web browser comprises an editor that edits on the client computer certain GUI Web Documents by adding and/or removing **applets** and links from the certain GUI Web documents. In this way, the GUI can be customized.

SUMMARY:

BSUM(17)

Moreover, the one or more **applets** of each GUI Web document are written in a platform independent programming language. As a result, the Web browser includes a virtual machine module that verifies the integrity of, interprets, and then executes on the client computer the **applets**. In the preferred embodiment, the platform independent programming language is the **Java** programming language and the virtual machine module is a **Java** virtual machine module.

DETDESC:

DETD(5)

The . . . commands issued by a user with the user input devices 112 and 113 in setting up the Web server to **download** the Web documents. And, it may be in response to requests received by the network interface 116 via the network interconnections 106 from users of the client computers 102 for **downloading** the Web documents to the client computers. In the preferred embodiment, the Web documents are HTML Web documents and the Web server is an HTTP server for **downloading** the HTML documents according to the HTTP.

DETDESC:

DETD(7)

The . . . 138 and a Web browser 140 which may be both loaded from the secondary memory 133. Alternatively, they may be **downloaded** loaded from one of the server computers 104 via the network interconnections 106. The primary memory also stores the Web documents 150 that have been either **downloaded** from the server computers 104 and/or loaded from the secondary memory. The operating system and Web browser are executed on. . . of the Web browser in response to commands issued by a user with the mouse 128 and/or keyboard 129 for **downloading** the Web documents 150 from the sever computers and/or loading them from the secondary memory. In the preferred embodiment, the Web browser is a **HotJava** (a trademark of Sun Microsystems) Web browser or **Java** compatible Web browser that includes a **Java** virtual machine module.

DETDESC:

DETD(12)

However, each GUI Web document 150 also includes one or more **applets** 156 that make it interactive. When the GUI Web document is displayed, the display manager 142 executes each **applet**. In response, each **applet** generates a corresponding interactive (IA) image 160 that is displayed by the display manager via the display driver of the. . . corresponding way when the displayed mouse arrow 144 or cursor 146 is over the displayed IA image. This invokes the **applet** that generated the IA image and the **applet** performs the corresponding operation on the client computer 102 and updates the IA image for display by the display manager.

DETDESC:

DETD(13)

Thus, . . . operations for inserting, deleting, cutting, and/or pasting text in a word processing document, this GUI Web document would include an **applet** 152 which generates an IA image 160 of the word processing document. And, when the IA image is acted on by the user with the mouse 128 and/or keyboard 129, the **applet** that generates the IA image is invoked so that the user is able to insert, delete, cut, and/or paste text. . . could have a toolbar with operations for selecting fonts and inserting page numbering. The GUI Web document would include an **applet** which generates an IA image listing the selectable fonts and an **applet** which generates an IA image providing various page numbering options. Thus, when these IA images are acted on by the user with the mouse 128 and/or keyboard 129, the **applets** that generates the IA images are invoked so that the user is able to select a font or page numbering option which may then be provided to the main **applet** generating the IA image of the word processing document for display in the word processing document.

DETDESC:

DETD(14)

As mentioned earlier, the Web browser 140 is, in the preferred embodiment, a **HotJava** Web browser or a **Java** compatible Web browser. Thus, in the preferred embodiment, the **applets** 156 are **Java applets** and the display manager 142 of the Web browser includes a **Java** virtual machine module for verifying, interpreting, and then executing on the client computer 102 the **Java applets**.

DETDESC:

DETD(20)

In order to edit a GUI Web document 150, a user **downloads** the GUI Web document in the manner described earlier. As was described earlier, this is done with the mouse 128. . .

DETDESC:

DETD(21)

Then, . . . own GUI. In response, the display manager 142 invokes the editor 149. The editor enables the user to edit the **downloaded** GUI Web document by inserting and/or deleting text 152, non-IA images 154, **applets** 156, and links 158.

DETDESC:

DETD(23)

In . . . adding and/or removing operations to and/or from certain existing GUI Web documents 150. This is done by adding and/or removing **applets** 156 from these GUI Web documents and adding and removing links 158 from the GUI Web documents which are linked. . .

CLAIMS:

CLMS(1)

What . . .
the user with the client computer and (b) while the link is being displayed on the client computer;
one or more **applets**, each of the one or more **applets** (a) generating a corresponding interactive image that is displayed on the client computer when the **applet** is executed and (b) being invoked to perform a corresponding one of the specific operations when the user acts on. . .
(b) displays the corresponding GUI Web document on the client computer by (i) executing each of the one or more **applets** of the corresponding GUI Web document on the client computer and displaying the corresponding interactive image on the client computer,. . .
links of the corresponding GUI Web document on the client computer, and
(c) invokes one of the one or more **applets** of the corresponding GUI Web document to perform the corresponding specific operation when the user acts on the corresponding interactive. . .

CLAIMS:

CLMS(3)

3. The GUI of claim 1 wherein:
the one or more **applets** of each of the GUI Web documents are written in a platform independent programming language; and

the Web browser includes a. . . in the client computer, verifies the integrity of, interprets, and then executes on the client computer the one or more **applets** of the particular GUI Web document.

CLAIMS:

CLMS(4)

4. The GUI of claim 3 wherein the platform independent programming language is the **Java** programming language and the virtual machine module is a **Java** virtual machine module.

CLAIMS:

CLMS(6)

6. . . . GUI of claim 5 wherein the editor enables the user to edit the selected GUI Web document by adding an **applet** to the one or more **applets** of the selected GUI Web document.

CLAIMS:

CLMS(7)

7. . . . the editor enables the user to edit the selected GUI Web document by removing one of the one or more **applets** of the selected GUI Web document.

CLAIMS:

CLMS(10)

10. . . .
the user with the client computer and (b) while the link is being displayed on the client computer;
one or more **applets**, each of the one or more **applets** (a) generating a corresponding interactive image that is displayed on the client computer when the **applet** is executed and (b) being invoked to perform a corresponding one of the specific operations when the user acts on. . .
computer
displaying the corresponding GUI Web document on the client computer by (i) executing each of the one or more **applets** of the corresponding GUI Web document on the client computer and displaying the corresponding interactive image on the client computer,. . . or more links of the corresponding GUI Web document on the client computer; and
invoking one of the one or more **applets** of the corresponding GUI Web document to perform the corresponding specific operation when the user acts on the corresponding interactive. . .

CLAIMS:

CLMS(12)

12. The method of claim 10 wherein
the one or more **applets** of each of the GUI Web documents are written in a platform independent programming language; and
the executing step includes, each. . . in the client computer, verifying the integrity of, interpreting, and then executing on the client computer the one or more **applets** of the particular GUI Web document with a virtual machine module.

CLAIMS:

CLMS(13)

13. The method of claim 12 wherein the platform independent programming language is the **Java** programming language and the virtual machine module is a **Java** virtual machine module.

CLAIMS:

CLMS(15)

15. . . . claim 14 wherein the enabling step includes enabling the user to edit the selected GUI Web document by adding an **applet** to the one or more **applets** of the selected GUI Web document.

CLAIMS:

CLMS(16)

16. . . . step includes enabling the user to edit the selected GUI Web document by removing one of the one or more **applets** of the selected GUI Web document.

CLAIMS:

CLMS(19)

19. . . .
the user with the client computer and (b) while the link is being displayed on the client computer;
one or more **applets**, each of the one or more **applets** (a) generating a corresponding interactive image that is displayed on the client computer when the **applet** is executed and (b) being invoked to perform a corresponding one of the specific operations when the user acts on. . .
(b) displays the corresponding GUI Web document on the client computer by (i) executing each of the one or more **applets** of the corresponding GUI Web document on the client computer and displaying the corresponding interactive image on the client computer,. . . links of the corresponding GUI Web document on the client computer, and (c) invokes one of the one or more **applets** of the corresponding GUI Web document to perform the corresponding specific operation when the user acts on the corresponding interactive. . .

CLAIMS:

CLMS(21)

21. The set of computer-readable modules of claim 19 wherein:
the one or more **applets** of each of the GUI Web documents are written in a platform independent programming language; and
the Web browser includes a. . . in the client computer, verifies the integrity of, interprets, and then executes on the client computer the one or more **applets** of the particular GUI Web document.

CLAIMS:

CLMS(22)

22. The set of computer-readable modules of claim 21 wherein the platform independent programming language is the **Java** programming language and the virtual machine module is a **Java** virtual machine module.

CLAIMS:

CLMS(24)

24. . . . modules of claim 23 wherein the editor enables the user to edit the selected GUI Web document by adding an **applet** to the one or more **applets** of the selected GUI Web document.

CLAIMS:

CLMS(25)

25. . . . the editor enables the user to edit the selected GUI Web document by removing one of the one or more **applets** of the selected GUI Web document.

US PAT NO: 5,761,421 [IMAGE AVAILABLE] L16: 3 of 3

ABSTRACT:
A system and method for establishing a peer-to-peer communication connection between computer programs from the same security domain, but executing in first and second computers, is disclosed. A first computer program, while executing in the first computer, sends a communication a message to the second computer, requesting a peer-to-peer communication connection. Upon receiving the message at said second computer, the second computer determines whether a second computer program meeting predefined criteria for establishing a peer-to-peer communication connection is executing in the second computer. If so, the second computer sends to the first computer a reply message accepting the request. After receipt of the reply message by the first computer, the requested peer-to-peer communication connection between the first and second computer programs is established. In a preferred embodiment, the predefined criteria for establishing a peer-to-peer communication connection is that the first and second computer programs be from the same server computer.
TITLE: System and method for secure peer-to-peer communication between **downloaded** programs
ASSIGNEE: **Sun Microsystems**, Inc., Palo Alto, CA (U.S. corp.)

SUMMARY:

BSUM(3)

The term "**applets**" is herein defined to mean computer programs and computer program fragments.

SUMMARY:

BSUM(4)

Due . . . computer programs in object-oriented computer systems are usually constrained to communicate in a client-server manner. For instance, in Sun Microsystems' **Java** virtual machine, when a method running in client computer requests an **applet** from a server computer, the browser program in the client computer marks the received **applet** to indicate the server from which the **applet** was received, and thereafter limits the information accessible to the **applet** to documents and other **applets** from the same server computer. Further, the **downloaded applet** is allowed by the **Java** virtual machine to open a communication channel to other **applets** on the server from which the **applet** was **downloaded**, but generally cannot open communication channels to **applets** in other computers.

SUMMARY:

BSUM(5)

It . . . different client computers to communicate securely. Typically, the two programs obtained from the same security domain will be two programs **downloaded** from the same server computer onto two different client computers. In many cases the two programs will be two copies of the same program **downloaded** onto two different client computers.

SUMMARY:

BSUM(6)

The basis for allowing such peer-to-peer connections is that sufficient security is provided when the communicating **applets** are both from the same server computer, because each **applet** would already have been allowed to open a communication channel to the server computer and therefore could have communicated indirectly with **applets** **downloaded** from the same server computer onto other client computers.

DETDESC:

DETD(2)

Referring . . . client computer includes a virtual machine, M1, M2, that provides the operating environment for executing a browser program such as **HotJava** (a product distributed by Sun Microsystems, Inc.) (not shown in FIG. 1) and for executing **Java** bytecode programs such as A1, A2 loaded through the use of the browser program. In the context of the present invention, the browser programs associated with virtual machines M1, M2 have been used to **download** program A from server S1, creating identical programs A1 and A2 and virtual machines M1 and M2, respectively.

DETDESC:

DETD(3)

Using standard client-server communication channels, it would be possible for **applet** A1 to communicate with **applet** A2 via the server S1 by setting up object class methods for use by the clients and server to create and maintain such communication channels. In other words, **applet** A1 could communicate securely with server S1 and then server S1 could communicate securely with **applet** A2, thereby creating a two stage connection between **applets** A1 and A2.

DETDESC:

DETD(4)

Typically, . . . that would be set aside specifically for the transmission of such messages. The communication sockets would be defined by the **applets** and server software so that messages received from an **applet** executing on a client computer would be received at the server at a communication socket that is monitored by a. . .

DETDESC:

DETD(5)

In the preferred embodiment, the **applets** and other programs being executed are primarily **Java** bytecode programs. The **Java** bytecode language is a "machine platform independent" programming language marketed by Sun Microsystems, Inc. **Java** bytecode programs are executed in conjunction with a bytecode program interpreter that forms a virtual machine. **Java** bytecode programs are designed so that they can be executed on any computer, regardless of the operating system and

computer hardware platform of the computer, so long as a **Java** bytecode program interpreter is present on the computer.

DETDESC:

DETD(9)
an Internet communications manager program 172, such as the **HotJava** browser program;

DETDESC:

DETD(10)
a **Java** bytecode program verifier 174 for verifying whether or not a specified program satisfies certain predefined integrity criteria;

DETDESC:

DETD(11)
a **Java** bytecode program interpreter 176 for executing application programs;

DETDESC:

DETD(16)

Referring . . . 3, prior to execution of the peer-to-peer communication protocol 196, at least two virtual machines M1 and M2 will have **downloaded** copies of the same application program A (copies A1 and A2) from a server computer S1 (steps 200, 202). Alternately, the two virtual machines M1 and M2 will have **downloaded** copies of two different application programs from the same server computer S1. For the purposes of explaining the invention, we will assume that the two **downloaded** programs are the same, but in some embodiments of the present invention different programs from the same server will establish. . .

DETDESC:

DETD(17)

In FIG. 3 we will assume that **applet** A1 in virtual machine M1 initiates the process of establishing a communication connection to **applet** A2 in virtual machine M2 by invoking a method that causes virtual machine M1 to send a message packet P1. . .

DETDESC:

DETD(18)

Virtual machine M2 receives the packet P1 and a control program that controls the operation of virtual machine M2 (e.g., the **HotJava** browser program executing in virtual machine M2) determines whether or not it meets predefined criteria for establishing a peer-to-peer communication connection (step 206). In a preferred embodiment the predefined criteria are that the receiving virtual machine has an **applet** whose server source matches the server source of the initiating **applet**. In addition, in the preferred embodiment there must be an **applet** or program executing in the receiving virtual machine that is set to receive messages on the socket or communication channel used by the initiating **applet**.

DETDESC:

DETD(19)

If . . . sends a reply packet P2 to virtual machine M1 accepting the request, and indicating that virtual machine M2 is running **applet** A2 rom server S1 (step 212).

DETDESC:

DETD(20)

In . . . where additional security is required, or where peer-to-peer communications are to be allowed only between identical copies of the same **applets**, the reply packet may include evidence that packet P2 was actually sent by virtual machine M2 and that M2 really has a copy of **applet** A2. For instance, in this alternate embodiment the reply packet P2 includes a copy of at least a predefined portion of a unique number associated with the **applet** A2.

DETDESC:

DETD(22)

Upon . . . reply packet P2, virtual machine M1 sends an acknowledgment message back to virtual machine M2, establishing a peer-to-peer connection between **applets** A1 and A2 (step 214). Thereafter, the two **applets** exchange messages and data (step 216) in accordance with the common security restrictions shared by the two **applets**.

CLAIMS:

CLMS(4)

4. . . . distributed computer system having first and second client computers and a server computer from which copies of computer programs are **downloaded** to said first and second client computers, said second client computer comprising:
a virtual machine with a communications procedure for. . .

CLAIMS:

CLMS(7)

7. . . . distributed computer system having first and second client computers and a server computer from which copies of computer programs are **downloaded** to said first and second client computers, said first client computer comprising:
a virtual machine with a communications procedure for. . .

=> d his

(FILE 'USPAT' ENTERED AT 07:08:26 ON 21 OCT 1998)

L1	0	S	SERVLET
L2	0	S	SERVLETS
L3	18	S	ACTIVEX
L4	11	S	SERVER AND L3
L5	8602	S	DOWNLOAD?
L6	9	S	L4 AND L5
L7	151323	S	SERVICE
L8	6	S	L7 AND L6
L9	951	S	SUN MICROSYSTEM? /ASN
L10	234	S	JAVA
L11	27	S	L9 AND L10
L12	14	S	HOTJAVA
L13	9	S	L11 AND L12
L14	7	S	L5 AND L13

```
L15          97 S APPLET#
L16           3 S L15 [redacted] L14

=> s (user or identif? or authentic?)(30a)(server or remote or code or user
or request? or client)

        276211 USER
        396857 IDENTIF?
         10657 AUTHENTIC?
          7948 SERVER
        220170 REMOTE
        129384 CODE
        276211 USER
         69953 REQUEST?
          4821 CLIENT
L17     125112 (USER OR IDENTIF? OR AUTHENTIC?)(30A)(SERVER OR REMOTE OR C
ODE
                OR USER OR REQUEST? OR CLIENT)

=> s l17(50a)(class? or service or program or load? or download? or interfac?
or parameter#)

        247653 CLASS?
        151323 SERVICE
        160020 PROGRAM
        626921 LOAD?
          8602 DOWNLOAD?
        242215 INTERFAC?
        263084 PARAMETER#
L18      37367 L17(50A)(CLASS? OR SERVICE OR PROGRAM OR LOAD? OR DOWNLOAD?
 OR
                INTERFAC? OR PARAMETER#)

=> s applet#(50a)(l18

UNMATCHED LEFT PARENTHESIS '50A)(L18'

=> s applet#(50a)(l18)

            97 APPLET#
L19         25 APPLET#(50A)(L18)

=> d 1-25
```

1. 5,826,031, Oct. 20, 1998, Method and system for prioritized downloading of embedded web objects; Jakob Nielsen, 395/200.63, 200.37, 200.49, 200.61 [IMAGE AVAILABLE]

2. 5,821,927, Oct. 13, 1998, Web browser display indicator signalling that currently displayed web page needs to be refereshed from remote source; Qing Gong, 345/335 [IMAGE AVAILABLE]

3. 5,819,220, Oct. 6, 1998, Web triggered word set boosting for speech interfaces to the world wide web; Ramesh Sarukkai, et al., 704/243, 240, 270 [IMAGE AVAILABLE]

4. 5,815,718, Sep. 29, 1998, Method and system for loading classes in read-only memory; Theron D. Tock, 395/705, 685, 710 [IMAGE AVAILABLE]

5. 5,815,683, Sep. 29, 1998, Accessing a remote cad tool server; Joe E. Vogler, 395/500, 200.47, 200.55 [IMAGE AVAILABLE]

6. 5,812,529, Sep. 22, 1998, Method and apparatus for network assessment; Paul G. Czarnik, et al., 370/245, 252; 395/200.33 [IMAGE

AVAILABLE]

7. 5,809,247, Sep. 15, 1998, Method and apparatus for guided touring of internet/intranet websites; John A. Richardson, et al., 395/200.48, 200.32, 200.59 [IMAGE AVAILABLE]

8. 5,805,829, Sep. 8, 1998, Process for running applets over non-IP networks; Geoffrey Alexander Cohen, et al., 395/200.32, 200.34, 200.49, 200.59 [IMAGE AVAILABLE]

9. 5,805,442, Sep. 8, 1998, Distributed interface architecture for programmable industrial control systems; Kenneth C. Crater, et al., 364/138; 340/825.07; 345/346; 364/131; 395/200.49, 200.5, 200.58 [IMAGE AVAILABLE]

10. 5,802,530, Sep. 1, 1998, Web document based graphical user interface; Arthur A. Van Hoff, 707/513; 345/335; 395/200.33, 682 [IMAGE AVAILABLE]

11. 5,802,388, Sep. 1, 1998, System and method for correction and confirmation dialog for hand printed character input to a data processing system; John Mark Zetts, et al., 707/541 [IMAGE AVAILABLE]

12. 5,796,952, Aug. 18, 1998, Method and apparatus for tracking client interaction with a network resource and creating client profiles and resource database; Owen Davis, et al., 395/200.54 [IMAGE AVAILABLE]

13. 5,784,553, Jul. 21, 1998, Method and system for generating a computer program test suite using dynamic symbolic execution of JAVA programs; Adam K. Kolawa, et al., 395/183.14 [IMAGE AVAILABLE]

14. 5,784,539, Jul. 21, 1998, Quality driven expert system; Frederick P. Lenz, 706/45 [IMAGE AVAILABLE]

15. 5,778,356, Jul. 7, 1998, Dynamically selectable language display system for object oriented database management system; William C. Heiny, 707/2, 103, 104 [IMAGE AVAILABLE]

16. 5,774,666, Jun. 30, 1998, System and method for displaying uniform network resource locators embedded in time-based medium; Michael J. Portuesi, 395/200.48; 348/473; 395/200.49 [IMAGE AVAILABLE]

17. 5,768,510, Jun. 16, 1998, Object-oriented system, method and article of manufacture for a client-server application enabler system; Sheri L. Gish, 395/200.33, 200.32, 200.48 [IMAGE AVAILABLE]

18. 5,757,925, May 26, 1998, Secure platform independent cross-platform remote execution computer system and method; Yaroslav Faybishenko, 380/49; 706/47 [IMAGE AVAILABLE]

19. 5,754,830, May 19, 1998, Server and web browser terminal emulator for persistent connection to a legacy host system and method of operation; Thomas H. Butts, et al., 395/500; 370/466, 469; 395/200.33, 200.48, 200.57, 285, 680 [IMAGE AVAILABLE]

20. 5,742,768, Apr. 21, 1998, System and method for providing and displaying a web page having an embedded menu; Giuseppe Gennaro, et al., 1/1 [IMAGE AVAILABLE]

21. 5,729,594, Mar. 17, 1998, On-line secured financial transaction system through electronic media; Edwin E. Klingman, 379/93.12; 705/26 [IMAGE AVAILABLE]

22. 5,706,502, Jan. 6, 1998, Internet-enabled portfolio manager system and method; Jill Paula Foley, et al., 707/10; 395/682 [IMAGE AVAILABLE]

23. 5,691,897, Nov. 2[redacted]1997, Motion control systems; [redacted]vid W. Brown, et al., 364/167.02; 318/568.1; 364/191 [IMAGE AVAILABLE]

24. 5,572,648, Nov. 5, 1996, System for simultaneously displaying a static tool palette having predefined windowing tool functions and a dynamic tool palette which changes windowing tool functons in accordance with a context of an executed application program; Farzad Bibayan, 345/340 [IMAGE AVAILABLE]

25. 5,548,745, Aug. 20, 1996, Method and apparatus for designating context in a windowing environment; Alistair Egan, et al., 395/500; 364/281.7, 282.3, 286.3, DIG.1 [IMAGE AVAILABLE]

=> d 1-25 fd,rel,as,ab,kwic

US PAT NO: 5,826,031 [IMAGE AVAILABLE] L19: 1 of 25
DATE FILED: Jun. 10, 1996
ASSIGNEE: Sun Microsystems, Inc., Palo Alto, CA (U.S. corp.)

ABSTRACT:
In the preferred embodiment of the present invention, the method retrieves a web file and sorts one or more information element references according to a priority attribute associated with each reference. After ranking the information element references by priority, the method then retrieves each information element in the order that their references were ranked. After receiving an information element, a user acts upon the information element in the appropriate manner. Typical objects, and their actions, would include displaying an image file, executing an applet, or playing an audio file. By using the steps of the preferred method, more important objects are retrieved before less important objects, thus allowing the user to act upon the more important objects sooner than the user could have acted on the important objects using methods available in the prior art.

SUMMARY:

BSUM(8)

Embodiments . . . referenced in an HTML file. For example, a web object may be, but is not limited to, text, a graphical **user interface** element, an image file, an audio file, an **applet**, or other computer **code**. "Acting on" the information element typically includes, but is not limited to, displaying the text, displaying the graphical **user interface** element, displaying the image file, playing the audio file, executing the **applet**, or executing other computer **code**.

US PAT NO: 5,821,927 [IMAGE AVAILABLE] L19: 2 of 25
DATE FILED: Jul. 25, 1996
ASSIGNEE: International Business Machines Corporation, Armonk, NY (U.S. corp.)

ABSTRACT:
Network browser applications are improved by providing visual status indications informing users that currently displayed pages are one of: old (outdated), partly old or new. Conventional browser applications load old or partly old page information from a cache and new information from a (usually remote) server to which the browser links via a network. A user expecting to view only new information (e.g. informant that might be useless if out of date) is alerted by present status indications to request the browser to reload the entire page; which the user can do by operating a reload selector/icon conventionally presented by the browser. Various alternative status indications are shown, along with potential

associations of such with a reload selector button (or equivalent icon).

DETDESC:

DETD(6)

This problem is complicated presently by the possibility that some of the current page information may be **loaded** partly from a **server** and partly from cache. For instance, where part of a page is a form completable by the viewer/**user**, information in that form may be accompanied by an "**applet**" used to control interaction between the **user**'s computer and the **remote** origin **server**, and other information on the same page may subject to frequent change at the origin **server**. In that situation, the browser may retrieve the form and **applet** from cache and the other/changeable information from either cache or the origin server. So this type of practice can cause. . .

US PAT NO: 5,819,220 [IMAGE AVAILABLE] L19: 3 of 25
DATE FILED: Sep. 30, 1996
ASSIGNEE: Hewlett-Packard Company, Palo Alto, CA (U.S. corp.)

ABSTRACT:
A computer system for user speech actuation of access to stored information, the system including a central processing unit, a memory and a user input/output interface including a microphone for input of user speech utterances and audible sound signal processing circuitry, and a file system for accessing and storing information in the memory of the computer. A speech recognition processor operating on the computer system recognizes words based on the input speech utterances of the user in accordance with a set of language/acoustic model and speech recognition search parameters. Software running on the CPU scans a document accessed by a web browser to form a web triggered word set from a selected subset of information in the document. The language/acoustic model and speech recognition search parameters are modified dynamically using the web triggered word set, and used by the speech recognition processor for generating a word string for input to the browser to initiate a change in the information accessed.

DETDESC:

DETD(13)

In . . . here at the local machine 12 will be using speech input via microphone and digital signal processing (DSP) as an **interface** to access the information on computer(s) 14 across the network 16. The user interacts with the local machine with the same **interface** as always (a netscape/mosaic web browser) depicted as WWW Browser 20 in FIG. 1. The only constraint in our implementation is that the **client** browser 20 can interpret html documents 22, and can execute java **applets**. In order to move from the current page to a different one, the **user** simply voices the highlighted words of the http link of interest.

US PAT NO: 5,815,718 [IMAGE AVAILABLE] L19: 4 of 25
DATE FILED: May 30, 1996
ASSIGNEE: Sun Microsystems, Inc., Mountain View, CA (U.S. corp.)

ABSTRACT:
A method and system for providing an executable module having an address space for storing program data that is to reside in a read-only storage medium and an address space for storing program data that is to reside in a random access memory is herein described. The executable module represents Java classes that are structured for dynamic class loading. A static class loader is used to modify the class structure to accommodate static loading. The static class loader also identifies methods that contain unresolved symbolic references and data that varies during the

execution of the module. These methods and data are identified in order to place them in the address space that resides in the random access memory. The static loader is beneficial in a distributed computing environment having a client computer that has little or no secondary storage thereby requiring applications to run entirely in random access memory. By utilizing a read-only memory to store statically loadable classes, the random access memory is left available for other uses.

DETDESC:

DETD(2)

The . . . such a client computer. Preferably, the application is a browser that is used to import Java content, such as Java **applets**, from one or more server computers. Typically, the browser is an interpreted **program** module that retrieves Web documents utilizing a HyperText Transfer Protocol (HTTP) to access one or more Web pages formatted as HyperText Markup Language (HTML) documents from a **server** acting as a Web site. The HTML documents are interpreted and presented to the **user** associated with the **client** computer. Often, the HTML documents embed **applets**. An **applet** is a executable module represented as a Java **class**. The browser **loads** in the **applet** and its associated **classes** in order to execute the **applet**.

DETDESC:

DETD(9)

In an embodiment, one or more **server** computers act as Web sites containing a repository of HTML documents containing Java content or **applets**. The **client** computer executes a browser that provides a **user** associated with the **client** computer with access to the HTML documents available from the **server** computer. Referring to FIG. 1, a **server** computer typically includes one or more processors 112, a communications **interface** 116, a **user interface** 114, and memory 110. Memory 110 stores:

US PAT NO: 5,815,683 [IMAGE AVAILABLE] L19: 5 of 25
DATE FILED: Nov. 5, 1996
ASSIGNEE: Mentor Graphics Corporation, Wilsonville, OR (U.S. corp.)

ABSTRACT:
An access facilitator is programmed to provide access service for facilitating remote client access to computer-aided design (CAD) tools. The access service includes services for accepting an access connection from a client, obtaining an internetworking address of the client, receiving an access request from the client, and routing the access request including the internetworking address to a CAD tool, resulting in the CAD tool directly responding to the client.

DETDESC:

DETD(11)

For . . . illustrated embodiment, software environment 50 of access facilitator 14 (FIG. 5) includes operating system 52, http server 54 and access **service** 56. Operating system 52 also includes graphics, interprocess communication, as well as network communication services, including in TCP/IP communication services. A particular example of operating system 52 is Windows NT.TM. of Microsoft Corp. of Redmond, Wash. Http-**server** 54 is known in the art. Http **server** 54 has access in particular to access control homepage 51, which includes a selection for a **user** of **client** 12 to initiate the access process. Http **server** 54 also has access to access connect **applet** 53 to be provided to **client** 12 in response to **client** 12 initiating the

access process. Access **service** 56 including its monitor **service** will be described in further detail below.

DETDESC:

DETD(13)

Referring . . . and a connection port of access facilitator 14. In other embodiments, these information are hard coded. Upon prompting, access connect **applet** 53 waits for the prompted information to be entered and an indication from the **user** to submit the entered information, step 86. Upon receiving the indication to submit the **client** information (e.g. the **user** clicking a "submit" button), access connect **applet** 53 establishes an access connection to access facilitator 14, and submits the entered information to access **service** 56, step 88.

US PAT NO: 5,812,529 [IMAGE AVAILABLE] L19: 6 of 25
DATE FILED: Nov. 12, 1996
ASSIGNEE: LanQuest Group, Fremont, CA (U.S. corp.)

ABSTRACT:
A system and method are disclosed for acquiring network performance data. A mission server is connected to a network, and is operative to interface with Clients to define and receive requests for a mission. The mission as defined includes operations that require participation in the network by devices connected to a plurality of segments at a plurality of locations within the network. A plurality of sentries are provided on devices connected to the segments of the network at locations within the network so that the devices are operative to participate in the network from the segments of the network at their locations. The sentries are then operative to support the mission by participating in the network through the devices. A request for a mission is received at the mission server and the mission is communicated from the mission server to the sentries required to execute the mission. The operations of the mission are executed by the sentries and the results of the operations are communicated from the sentries to the mission server. The result of the mission is determined from the results of the operations.

DETDESC:

DETD(47)

FIG. . . . FIG. 1. The process begins at step 300. In step 302, the Client 110 connects to the mission server and **downloads** information needed to select a mission. The mission is defined in step 304 in cooperation with the mission server. In one embodiment, this is accomplished simply by making selections from web pages provided by the mission **server**. In another embodiment, the **Client** 110 **downloads** Java **applets** and uses those **applets** to assist the **user** to define the mission. The mission is **requested** in step 306.

US PAT NO: 5,809,247 [IMAGE AVAILABLE] L19: 7 of 25
DATE FILED: Jul. 22, 1996
ASSIGNEE: Intel Corporation, Santa Clara, CA (U.S. corp.)

ABSTRACT:
The present invention for guided touring of websites includes a web tour director programmed onto a web server for connecting a client system to a number of web sites in accordance with a web tour stop vector identifying the web sites as tour stops of a web guided tour. The present invention further includes a media rendering function also programmed onto the web server for rendering on the client system, one or more corresponding media for each of the web sites, for at least a portion of the time while the web site is connected to the client system.

CLAIMS:

CLMS(17)

17. . . .
stop definitions, each of which includes a duration of a stay at each tour stop;
b) building a first graphical end **user interface** with the first navigation **applet** for facilitating dynamic **user** modifications to the web tour; and
c) creating a web tour director **applet** upon building the first graphical end **user interface**, resulting in the web tour director **applet** being provided to the web touring station from the tour operator website.

CLAIMS:

CLMS(18)

18. . . . method as set forth in claim 17, wherein
step (a) further comprises the tour operator website providing a second navigation **applet** to the web touring station; and
the method further comprises the steps of (d) the second navigation **applet** building a second graphical end **user interface** for facilitating dynamic **user** modifications to the web tour, and (e) upon building the second graphical end **user interface**, the second navigation **applet** subsuming the role of the first navigation **applet**.

US PAT NO: 5,805,829 [IMAGE AVAILABLE] L19: 8 of 25
DATE FILED: Oct. 1, 1996
ASSIGNEE: International Business Machines Corp, Armonk, NY (U.S. corp.)

ABSTRACT:
A method and apparatus for allowing applets to be executed natively over a non-IP network. The method and apparatus provide an applet loader that initiates the applet download, services the class faults that are encountered and allows calls to non-IP APIs without compromising the security mechanisms of Java running on TCP/IP. This allows applets to be run without web browsers or web servers.

SUMMARY:

BSUM(3)

Initially, . . . Sun Microsystems), an internet-capable interpreter (also called a Virtual Machine, or VM). With Java, web pages can include programs, called **applets**. When a browser **downloads** an **applet**-containing page, the browser extracts the **applet** from the page and submits it to the Java virtual machine for execution. Thus, a **program** or **applet** can execute on a **user**'s machine without the **user** explicitly installing it.

DETDESC:

DETD(6)

As shown in the flow chart of FIG. 1a, an **applet loader** is used to execute the following processes. First, at 101, the **user** is prompted for the URL of the desired **applet**. The application next reads the URL name that is entered by the **user** 102. This completes the process of gathering the input from the **user**. The URL is then parsed by the application 103 to separate the **server** name and the

applet name from the information input by the **user**. Next, a connection is established with the **server** containing the desired **applet** 104. This connection is not an IP connection. In the preferred embodiment, it is an SNA connection using LU 6.2 protocol. The preferred embodiment uses ACOPY on the client and AFTPD on the server, both published by IBM. The **applet loader** establishes the connection by issuing a Java "native call" to ACOPY.

CLAIMS:

CLMS(1)

What is claimed is:

1. A method for executing an **applet** represented as **class** files from a computer workstation over a non-IP network, said method comprising the steps of:
a **user** inputing a name for an **applet** to be invoked;
parsing a **server** name and an **applet** name from the input name;
establishing a connection with said named **server** if said **server** is not the workstation from which the **user** input came;
issuing a file transfer **request** to said **server** for a first **class** file of said **applet**;
receiving said first **class** file from said **server**;
submitting said first **class** file to a virtual machine, thereby executing said **applet**;
wherein said **applet** requires additional **class** files causing said virtual machine to issue a `**class** fault`; and
in response to said **class** fault, said workstation re-establishes a connection with said named server;
issues a file transfer request to said server for said additional. . .

CLAIMS:

CLMS(3)

3. A computer workstation comprising:
a non-IP connection to a computer network;
means for allowing a **user** to input a name for an **applet** to be invoked across said non-IP network;
means for parsing said name to **identify** an **applet** name and a **server** name;
means for establishing a non-IP connection to said **server** parsed from said name if said **server** is not said computer workstation;
means for issuing a file transfer request to said server for a first **class** file of said **applet** parsed from said name;
means for receiving said first **class** file at said workstation from said server;
means for executing said **applet** from said workstation across said non-IP network;
means for processing a `class fault` indicating said **applet** requires additional class files from said server;
means for re-establishing a non-IP connection to said server;
means for issuing a file transfer. . .

US PAT NO: 5,805,442 [IMAGE AVAILABLE] L19: 9 of 25
DATE FILED: May 30, 1996
ASSIGNEE: Control Technology Corporation, Hopkinton, MA (U.S. corp.)

ABSTRACT:
An integrated control system comprises one or more controllers each equipped to perform a control function and to gather data (ordinarily from sensors) relevant to the control function. Each controller contains computer storage means, such as computer memory, for storing the relevant data and instructions, associated with the data, for causing a remote

computer to generate a visual display incorporating the data in a predetermined format; and a communication module for establishing contact and facilitating data interchange with the remote computer. The remote computer, in turn, also includes a communication module compatible with the controller-borne module, and which enables the remote computer to download the data and associated instructions from one or more controllers. The remote computer also includes a facility for processing the instructions to create a user interface encoded by the instructions, and which incorporates the data. In this way, controller data is coupled to instructions for displaying that data, and this totality of information is continuously accessible, on a freely selective basis, to the remote computer.

DETDESC:

DETD(23)

Security becomes particularly important if the controller-based web pages allow **client** computer 50 not only to access data, but to modify it as well. For example, while "read-only" access to control data suffices to inform the **client user** of the state of a controlled machine or process, the **user** cannot, if limited to such access, influence the operation of the controller. It may prove desirable, therefore, to allow an appropriately authorized **client** to directly modify control **parameters** (which may, for example, be stored on a restricted-access web page) that determine the operation of the controller and, hence, the controlled machine or process. Indeed, a controller-based **applet** invoked by the **user**'s interaction with one of the controller's web pages can permit the remotely situated **client user** to operate the controller hardware--for example, causing the controller to execute a reset routine that restarts automated equipment following shutdown,. . .

US PAT NO: 5,802,530 [IMAGE AVAILABLE] L19: 10 of 25
DATE FILED: Jul. 1, 1996
ASSIGNEE: Sun Microsystems, Inc., Palo Alto, CA (U.S. corp.)

ABSTRACT:
A Web document based GUI for use on a client computer that is networked with server computers. The GUI enables a user of the client computer to initiate specific operations that are performed on the client computer and that define a particular application. The GUI comprises GUI Web documents and a Web browser. Each GUI Web document is located at the client computer or one of the server computers and comprises one or more links and one or more **applets**. Each link provides a link to a corresponding GUI document when selected by the user with the client computer while being displayed on the client computer. Each respective **applet** generates, when executed on the client computer, an interactive image that is displayed on the client computer. The **user** can initiate a respective operation (i.e., one of the GUI's specific operations) by acting on the interactive image with the **client** computer to invoke the respective **applet** to perform the respective operation on the **client** computer. The Web browser runs on the **client** computer and, each time a displayed link of a displayed GUI Web document has been selected by the **user** with the **client** computer, **loads** in, if not already **loaded**, and displays on the **client** computer the corresponding GUI Web document. The Web browser displays the corresponding GUI web document by executing each of the one or more **applets** of the corresponding GUI Web document and displaying on the client computer the corresponding interactive image and by displaying on the client computer the one or more links of the corresponding GUI Web document. The Web browser comprises an editor that edits on the client computer certain GUI Web Documents by adding and/or removing applets and links from the certain GUI Web documents. In this way, the GUI can be customized.

ABSTRACT:
A . . . at the client computer or one of the server computers and comprises one or more links and one or more **applets**. Each link provides a link to a corresponding GUI document when selected by the user with the client computer while being displayed on the client computer. Each respective **applet** generates, when executed on the client computer, an interactive image that is displayed on the client computer. The **user** can initiate a respective operation (i.e., one of the GUI's specific operations) by acting on the interactive image with the **client** computer to invoke the respective **applet** to perform the respective operation on the **client** computer. The Web browser runs on the **client** computer and, each time a displayed link of a displayed GUI Web document has been selected by the **user** with the **client** computer, **loads** in, if not already **loaded**, and displays on the **client** computer the corresponding GUI Web document. The Web browser displays the corresponding GUI web document by executing each of the one or more **applets** of the corresponding GUI Web document and displaying on the client computer the corresponding interactive image and by displaying on. . .

SUMMARY:

BSUM(9)

Another . . . virtual machine module determines whether Java applets conform to predefined stack usage and data usage restrictions to ensure that Java **applets** cannot overflow or underflow the virtual machine module's stack and utilize only data of known data types. As a result, Java **applets** cannot create object pointers and generally cannot access system resources other than those resources which the **user** explicitly grants it permission to use. Consequently, when Java **applets** are **downloaded** to a **client** computer, a Web browser that is running on the **client** computer and has a Java virtual machine module will be able to verify and then execute the **downloaded applets**.

SUMMARY:

BSUM(14)

The Web browser runs on the **client** computer and, each time a displayed link of a displayed GUI Web document has been selected by the **user** with the **client** computer, **loads** in, if not already **loaded**, and displays on the **client** computer the corresponding GUI Web document. The Web browser displays the corresponding GUI web document by executing each of the one or more **applets** of the corresponding GUI Web document and displaying on the client computer the corresponding interactive image and by displaying on. . .

CLAIMS:

CLMS(1)

What . . .
the user with the client computer and (b) while the link is being displayed on the client computer;
one or more **applets**, each of the one or more **applets** (a) generating a corresponding interactive image that is displayed on the client computer when the **applet** is executed and (b) being invoked to perform a corresponding one of the specific operations when the **user** acts on the corresponding interactive image with the **client** computer; and
a Web browser that runs on the **client** computer and that, each time a selected one of the one or more links of a displayed one of the GUI Web

documents has been selected by the **user** with the **client** computer, (a) **loads**, if not already **loaded**, the corresponding GUI Web document in the **client** computer, (b) displays the corresponding GUI Web document on the **client** computer by (i) executing each of the one or more **applets** of the corresponding GUI Web document on the client computer and displaying the corresponding interactive image on the client computer,. . .

CLAIMS:

CLMS(10)

10. . . .
the user with the client computer and (b) while the link is being displayed on the client computer;
one or more **applets**, each of the one or more **applets** (a) generating a corresponding interactive image that is displayed on the **client** computer when the **applet** is executed and (b) being invoked to perform a corresponding one of the specific operations when the **user** acts on the corresponding interactive image with the **client** computer; and
each time a selected one of the one or more links of a displayed one of the GUI Web documents has been selected by the **user** with the **client** computer;
loading, if not already **loaded**, the corresponding GUI Web document in the **client** computer
displaying the corresponding GUI Web document on the **client** computer by (i) executing each of the one or more **applets** of the corresponding GUI Web document on the client computer and displaying the corresponding interactive image on the client computer,. . .

CLAIMS:

CLMS(19)

19. . . .
the user with the client computer and (b) while the link is being displayed on the client computer;
one or more **applets**, each of the one or more **applets** (a) generating a corresponding interactive image that is displayed on the client computer when the **applet** is executed and (b) being invoked to perform a corresponding one of the specific operations when the **user** acts on the corresponding interactive image with the **client** computer; and
a Web browser that runs on the **client** computer and that, each time a selected one of the one or more links of a displayed one of the GUI Web documents has been selected by the **user** with the **client** computer, (a) **loads**, if not already **loaded**, the corresponding GUI Web document in the **client** computer, (b) displays the corresponding GUI Web document on the **client** computer by (i) executing each of the one or more **applets** of the corresponding GUI Web document on the client computer and displaying the corresponding interactive image on the client computer,. . .

US PAT NO: 5,802,388 [IMAGE AVAILABLE] L19: 11 of 25
DATE FILED: Dec. 19, 1996
REL-US-DATA: Continuation of Ser. No. 434,239, May 4, 1995, abandoned.
ASSIGNEE: IBM Corporation, Armonk, NY (U.S. corp.)

ABSTRACT:
A data processing system corrects handprinted character input represented as a sequence of points described by a writing path of a pointing device. The system receives a writing path signal from a pointing device in the data processing system, describing a first hand printed character input. The system performs character recognition on the writing path signal to

provide a first character string. The system then displays the first character string in an edit pad area. The system then receives a correcting writing path signal in the edit pad area from the pointing device, describing a correction to the first hand printed character input. The system then performs character recognition on the correcting writing path signal to provide a second character string.

DETDESC:

DETD(9)

In . . . layer 350 and these pen aware applications really have no need for the compatibility module 345. Small application programs, or **Applets** 305 are short applications that are included with the Pen For OS/2 product, that help users with productivity aids such as a pop-up hand writing window, or a pop-up sketch pad that allows the **user** to doodle or write into a bit map. Pen PM installation 300 is a **program** the **user** invokes to install the Pen For OS/2 product on the computer and the setup objects 310 are an adjunct to. . .

US PAT NO: 5,796,952 [IMAGE AVAILABLE] L19: 12 of 25
DATE FILED: Mar. 21, 1997
ASSIGNEE: Dot Com Development, Inc., New York, NY (U.S. corp.)

ABSTRACT:
A method for monitoring client interaction with a resource downloaded from a server in a computer network includes the steps of using a client to specify an address of a resource located on a first server, downloading a file corresponding to the resource from the first server in response to specification of the address, using the client to specify an address of a first executable program located on a second server, the address of the first executable program being embedded in the file downloaded from the first server, the first executable program including a software timer for monitoring the amount of time the client spends interacting with and displaying the file downloaded from the first server, downloading the first executable program from the second server to run on the client so as to determine the amount of time the client interacts with the file downloaded from the first server, using a server to acquire client identifying indicia from the client, and uploading the amount of time determined by the first executable program to a third server. The first executable program may also monitor time, keyboard events, mouse events, and the like, in order to track choices and selections made by a user in the file, and may execute upon the occurrence of a predetermined event, as well as monitoring or determining the amount of information downloaded by the client. The monitored information and client identifying indicia is stored on a database in a server for use in analysis and for automatically serving out files assembled according to user interests and preferences.

DETDESC:

DETD(36)

The . . . the tracked information (S610A, S610B). In step S610A, the second CGI script may obtain any information acquired by the tracking **program** (i.e., the JAVA **applet**), as well as **client identifying** indicia transmitted by the **client**, such as in the HTTP **request** header. This information can be stored in a database on **Server** B. If necessary, the information stored by both scripts may be combined into one more complete databases.

DETDESC:

DETD(40)

In . . . note of the current time (S708). Thereafter, the applet contacts the Server A and, if security restrictions allow it, the **applet** queries the Server A for the page it is embedded in, determines its size, as well as the URLs of. . . video), and requests header information about these resources in order to determine their size (S709). In this case, the tracking **program** may determine the size of the fully rendered Web page, (i.e., the number of bits that must be **downloaded** in order to fully render the Web page). If the tracking **program** is part of a larger embedded application that displays information **downloaded** from a **server** (such as a live news feed **applet**), the tracking **program** can also monitor the amount of information **downloaded** and displayed by the **applet**. Before or as the **user** leaves the Web page (S710), the tracking **program** can transmit this information to **Server** B for storage and analysis (S711, S711A, S711B). In this manner, it is possible to build a database of accurate. . . how often different pages of a Web site are requested, how long they are displayed, and how much information was **downloaded**. This information would be of use to Web site administrators in order to judge the popularity of different Web pages,. . .

DETDESC:

DETD(41)

In . . . according to the amount of information displayed, either according to bit size or time, or both. Imagine that the tracking **program** is attached to a live feed **applet**. The tracking **program** monitors the time the information is displayed and the amount of bits **downloaded** and automatically transmits this information back to a **server** when the **user** leaves. Together with the **user**'s ID (**client** and network), and billing information that the **user** was previously **requested** to enter, it is possible to determine the correct charge for the **user**. Similarly, a **user** could be charged and billed for time spent on a Web page, as well as amount of information **downloaded** by him or her.

DETDESC:

DETD(44)

In addition, while the preferred embodiments have been described in connection with JAVA **applets** that are executable on a **client**, the tracking of **user** interaction may be accomplished by a **client** executable **program** written in a language other than JAVA. For example, the teachings of the present invention may be accomplished using Active-X components in conjunction with the Internet Explorer Web browser. In addition, the tracking **program** need not be a **program** that executes on the client computer. For example, the tracking program may comprise a CGI script located on a server.. . .

US PAT NO: 5,784,553 [IMAGE AVAILABLE] L19: 13 of 25
DATE FILED: Apr. 30, 1997
REL-US-DATA: Continuation-in-part of Ser. No. 599,719, Feb. 12, 1996, which is a continuation-in-part of Ser. No. 587,208, Jan. 16, 1996, abandoned.
ASSIGNEE: Parasoft Corporation, Monrovia, CA (U.S. corp.)

ABSTRACT:
A method and system for generating a test suite for a computer program written in the JAVA programming language. The JAVA program comprises program statements and program variables represented as JAVA source code and compiled by a JAVA compiler into JAVA bytecodes, including at least one input statement having one or more input variables, that are grouped into code blocks and stored in a program database. The test suite comprises sets of inputs. Each of the sets of inputs corresponds to a pth

in the program. The program statements corresponding to the candidate code block are read from the program database. Each of the input variables for each input statement and each of the program variables that depend on them are represented in symbolic form as a symbolic memory value and transforming each program statement dependent on such an input variable into a symbolic expression. A trial set of inputs for each of the input statements is created by finding a solution to the symbolic expression obtained using dynamic symbolic execution. The trial set of inputs are stored into the test suite if coverage of the candidate code block was obtained. A dynamic symbolic execution consists of a symbolic execution of the program performed along the path that corresponds to the trial set of actual inputs. The first input to the program is generated randomly. From that first input, inputs satisfying any coverage criteria can be obtained by performing the above procedure iteratively.

DETDESC:

DETD(156)

A JAVA Platform is a software platform for delivering and running highly interactive, dynamic, and secure **applets** and applications on networked computer systems. **Applets** are programs that require a browser **program** to run. An <**applet**> tag is embedded in a World Wide Web (WWW) hyper-text markup language (HTML) file and **identifies** a **program** to be run. When that file (i.e., that page) is accessed by a **user**, either over the Internet or a corporate intranet, the **applet** automatically **downloads** from a **server** and runs on the **client** system. Applications are computer programs that do not require a browser to run and have no built-in **downloading** mechanism. When an application is called, it is executed. Applications perform traditional desktop tasks, such as word processing, spreadsheet, or graphics applications, and games. **Applets** and applications originate as JAVA source code files 200. These JAVA source code files have a ".java" suffix.

US PAT NO: 5,784,539 [IMAGE AVAILABLE] L19: 14 of 25
DATE FILED: Nov. 26, 1996
ASSIGNEE: Client-Server-Networking Solutions, Inc., White Plains, NY (U.S. corp.)

ABSTRACT:
A quality driven expert system maps answers input by a user in a user questionnaire into a set of qualities descriptive of a desired output entity. A library of output entities, such as network architectures is provided, each described by values for various ones of the qualities. The qualities derived by the expert system from the user's answers are matched against the qualities of the various output entities in the library to determine which entities match the specified qualities. A score for each output entity is determined by an associative matching algorithm as a function of the degree of match between each entity and the specified qualities, along with an explanation of the degree of match between each specified quality and the qualities of the output entity. A selected number of the highest scoring entities are returned to the user to further inspection and comparison. The expert system architecture may be applied in various application domains, one useful implementation of which is for the selection of network architectures for computer networks.
SYSTEM LIMITS EXCEEDED - DISPLAY ENDED

=> d l19 15-25 fd,re,as,ab,kwic

'RE' IS NOT A VALID FORMAT FOR FILE 'USPAT'
ENTER DISPLAY FORMAT (CIT):rel

US PAT NO: 5,778,356 [IMAGE AVAILABLE] L19: 15 of 25
REL-US-DATA: Continuation-in-part of Ser. No. 339,481, Nov. 10, 1994, and a continuation-in-part of Ser. No. 527,161, Sep. 12, 1995, and a continuation-in-part of Ser. No. 526,555, Sep. 12, 1995.

US PAT NO: 5,774,666 [IMAGE AVAILABLE] L19: 16 of 25

US PAT NO: 5,768,510 [IMAGE AVAILABLE] L19: 17 of 25

US PAT NO: 5,757,925 [IMAGE AVAILABLE] L19: 18 of 25

US PAT NO: 5,754,830 [IMAGE AVAILABLE] L19: 19 of 25

US PAT NO: 5,742,768 [IMAGE AVAILABLE] L19: 20 of 25

US PAT NO: 5,729,594 [IMAGE AVAILABLE] L19: 21 of 25

US PAT NO: 5,706,502 [IMAGE AVAILABLE] L19: 22 of 25

US PAT NO: 5,691,897 [IMAGE AVAILABLE] L19: 23 of 25

US PAT NO: 5,572,648 [IMAGE AVAILABLE] L19: 24 of 25

US PAT NO: 5,548,745 [IMAGE AVAILABLE] L19: 25 of 25

=> d l19 15-25 ab,kwic

US PAT NO: 5,778,356 [IMAGE AVAILABLE] L19: 15 of 25

ABSTRACT:
A dynamically selectable language display system for object oriented database management systems is disclosed. Class objects are provided having international string parameters that include a pointer to an international string list, the international string list including a language handle structure linked to a plurality of character strings in different languages. A handle manager is provided which is operative to select a character string corresponding to one of said plurality of character strings for display which corresponds to a dynamically selectable user specified language handle, whereby a representation of said class object may be simultaneously displayed to a plurality of users in different languages based upon a language handle individually selectable by each of said plurality of users.

DETDESC:

DETD(15)

FIG. . . . disk controller 147. Alternatively, some portion of such programs may consist of executable content such as Java script or Java **applets downloaded** from the wide area network 128, which may be the Internet 128. The **user** interacts with the system through the keyboard 115 and a mouse or similar graphical pointer 114 with the graphical **user interface** displayed on the CRT display 113. The API 143 may communicate with the knowledge base **server** 132 through a local area network 100, access to which is facilitated by a network controller 148, or through a wide area network 128, access to which is facilitated by a serial **interface** controller 151. Alternatively, an API 143 may communicate with a proxy server through the wide area network 128. An I/O. . .

US PAT NO: 5,774,666 [IMAGE AVAILABLE] L19: 16 of 25

ABSTRACT:

A system and method are provided for displaying a uniform network resource locator embedded in a time-based medium. In one embodiment, the time-based medium can be a movie file having an uniform network resource locator embedded by association with a track in the movie file. In another embodiment, the time-based medium can be a video signal having encoded information defining an embedded uniform network resource locator. An output for display is generated based upon the time-based medium where display of the output shows the embedded uniform network resource locator to a user such that the embedded uniform network resource locator is active during display of the output. The user is then allowed to activate the embedded uniform network resource locator. In response to activation by the user, the embedded uniform network resource locator is followed to retrieve a resource addressed by the embedded uniform network resource locator.

SUMMARY:

BSUM(6)

There are additional movie-type displays that can be created through the use of executable languages such as the use of JAVA **applets** to animate graphics. In addition, POINTCAST can be used to broadcast static screens over the public Internet or private intranets that are updated to provide a slide show presentation. Other conventional presentation software applications allow a **user** to build video into a presentation including DIRECTOR, available from MACROMEDIA, which allows a **user** to **program** a presentation which can include video. Relatively new technologies are also available that integrate common television with Internet web activity. . .

US PAT NO: 5,768,510 [IMAGE AVAILABLE] L19: 17 of 25

ABSTRACT:
An interprise computing manager in which an application is composed of a client (front end) program which communicates utilizing a network with a server (back end) program. The client and server programs are loosely coupled and exchange information using the network. The client program is composed of a User Interface (UI) and an object-oriented framework (Presentation Engine (PE) framework). The UI exchanges data messages with the framework. The framework is designed to handle two types of messages: (1) from the UI, and (2) from the server (back end) program via the network. The framework includes a component, the mediator which manages messages coming into and going out of the framework. The system includes software for a client computer, a server computer and a network for connecting the client computer to the server computer which utilize an execution framework code segment configured to couple the server computer and the client computer via the network, by a plurality of client computer code segments resident on the server, each for transmission over the network to a client computer to initiate coupling; and a plurality of server computer code segments resident on the server which execute on the server in response to initiation of coupling via the network with a particular client utilizing the transmitted client computer code segment for communicating via a particular communication protocol.

DETDESC:

DETD(72)

A . . . accordance with a preferred embodiment can execute on any processor with the Java interpreter/runtime (Java JDK) or JavaOS installed. Application **client** programs are developed in Java using a null application template that contains the necessary Java **classes** and methods for integration with a Graphical **User Interface** (GUI). The template includes Java **classes** which will allow the **client program** to communicate with the **server program**.

Scripts and tools for installing and deploying applications, include a generalized startup **applet** for application launch from a Web browser or **applet** viewer.

DETDESC:

DETD(82)

FIG. 6 illustrates the processing associated with application startup in accordance with a preferred embodiment. When an application is started, the **client** node 600 executes a startup **applet** 620 which first collects information about the **client** node 600 **user** and contacts the **server** node 610 via HTTP 602. The **server** node 610 has been extended to include a web **server** 630 for processing **requests** via HTTP 602 over the Web technologies in an Internet, Intranet or other network environment. The access layer 640 is called via a cgi-bin **interface** from the web **server** 630. The access layer 640 provides a framework so that the information about the **client** node 610 **user**, for example userid and password, can be used to call **server**-resident **authentication** services. Should **authentication** be successful, the access layer 640 uses the application name, which is also supplied by the startup **applet** 620, and invokes the app manager 650. The app manager 650 handles the application definitions installed on the server node. . .

DETDESC:

DETD(126)

Enterprise . . . and utilization of an application URL in accordance with a preferred embodiment. When the application URL 2615 is started, the **client** node 2600, through its browser 2605, executes startup **applet** 2610 to thereby collect information about the **client** **user** and contact the **server** node 2620. the **server** node 2620 includes a **server program** 2625, a database 2630, and access layer 2635, and application manager 2640, a web **server** 2645 and presentation engines 2650. The details concerning launch and utilization of application URL 2635 can be found in the. . .

DETDESC:

DETD(174)

The . . . application she wants to run. The URL is the address for a Web page that includes an ICE-T application startup **applet**. The Web page with the startup **applet** is **loaded** into the browser. The **applet** collects access information from the **user**. The **applet** contains the URL of the **server** holding the application components and the application name. This information is processed on the **server**. If the **user** name, password, and chosen application are authorized, the **server downloads** a Presentation Engine to the **user**'s node.

DETDESC:

DETD(181)

When . . . the server node using HTTP. The server manages this Web connection. ICE-T applications can be launched from a browser, an **applet** viewer, or as standalone applications. FIG. 26 illustrates the steps associated with launching an application URL in accordance with a preferred embodiment. On the **server** side, the ICE-T Access Layer (a cgi-bin executable) **authenticates** the **user** data. If the **authentication** succeeds, the Access Layer contacts the ICE-T Application Manager and the Application Manager starts the **server** **program** and initiates a network session.

DETDESC:

DETD(182)

The Application Manager **downloads** an HTML page with a startup **applet** for the application. When the **user** runs the startup **applet**, the Application Manager selects a compiled Presentation Engine and **downloads** an HTML page containing the **applet** tag for it to the **client** using HTTP. The compiled Presentation Engine includes a GUI and an instance of the client Communication Library and is ready. . .

DETDESC:

DETD(551)

When the **user** launches the stamp **applet**, the startup **applet** takes the name of the application from the **applet parameter**, the **user** name, and the password and sends these using HTTP to the Access Layer on the **server**.

DETDESC:

DETD(557)
startAppletDevIR.java defines a Send button and a **class** (sendBtn) that handles the **user**'s input. appTemplate.html includes default instructions for using Send. If you want to change the **user**'s interaction with the stamp **applet**, you would need to change the sendBtn **class** and the instructions in appTemplate. html.

DETDESC:

DETD(617)

Table Default Access Properties

Property	Default	Description
cgi.sub.-- bin.sub.-- location	/cgi -bin	If you have not created a link. . . Uses the scrambler key provided by ICE-T.
		Return value is used as a key to unscramble the **user** name and password sent when the **user** launches an application. If you change the scrambler key, change the scrambler key in the startup **applet** also.
sessionEnvironment		Sets up the calling environment for the **server program**. You can make calls to putenv ( ) to set environment variables whose values can be accessed with getenv ( ) from the server **program**.

DETDESC:

DETD(680)

The ICE-T application installation script (ice-app- install) generates a

default application configuration file (appConfigFile). When a **user** starts an application launching a startup **applet**, the Access Layer uses the Application Manager to look up values for **server** and **client program** locations and names in appConfigFile. Using the configuration file, the Application Manager generates an HTML wrapper for presenting Presentation Engines as **applets** in a Web browser execution environment. (See "Using Startup **Applets** and HTML Files" for more information about how to use startup **applets** for ICE-T applications.)

US PAT NO: 5,757,925 [IMAGE AVAILABLE] L19: 18 of 25

ABSTRACT:
A method of operating a computer system including a client computer with a user input mechanism and a display for displaying a user interface including user interface elements, an application server including an operating system and a user application configured to run under the operating system and a network interconnecting the client and the server. The method includes the steps of providing a user interface management program configured to run on the client and receive user events, accessing the application server over the network from the client, sending a request from the client to the server to run the application, running the application on the server to thereby generate user interface element definitions on the server and reporting user interface element definitions to the user interface management program over the network. The method also includes the steps of providing a set of user interface element rules accessible to the user interface management program, where the rules allow the user interface management program to filter user events and decide which can be initially handled on the client to update user interface elements on the client display and which will be forwarded to the server for initial processing and processing reported user interface element definitions in the user interface management program to display on the client user interface elements corresponding to the user interface element definitions.

US PAT NO: 5,754,830 [IMAGE AVAILABLE] L19: 19 of 25

ABSTRACT:
A computer network environment (10) allowing connection of a client system (36) to a legacy host system (18,19) using a server (26) is provided. The computer network environment (10) includes a legacy host system (18,19) having TCP/IP connectivity. The legacy host system (18,19) is operable to support a terminal session for access to the legacy host system (18,19). The computer network environment (10) also includes a server system (24) executing a client thread (28) under a server (26). The client thread (28) is operable to communicate with the legacy host system (18,19) across a persistent TCP/IP socket connection (30). The computer network environment (10) further includes a client system (36) executing an applet process (42) under a web browser (38). The applet process (42) is operable to communicate with the client thread (28) across another persistent TCP/IP socket connection (44) and is operable to provide a terminal session to a user of the client system (36). This terminal session is supported by a persistent connection allowing real-time bidirectional communication with the legacy host system (18,19).

SUMMARY:

BSUM(10)

According . . . client thread operable to communicate with a legacy host system across a persistent TCP/IP socket connection. The server also includes **applet** code operable to create an **applet** process executing under a web browser on a **client** system. When executed, the **applet** process is operable to communicate with the **client** thread across another persistent TCP/IP socket connection and to provide a